ĐẠI HỌC BÁCH KHOA HÀ NỘI
TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN: LƯU TRỮ VÀ XỬ LÝ DỮ LIỆU LỚN

Đề tài: **PHÂN TÍCH DỮ LIỆU CỜ VUA TỪ LICHESS**

Mã học phần: IT4931
Mã lớp: 154050
Nhóm sinh viên thực hiện: Nhóm 8

STT	Họ và tên	MSSV	Email
1	Tô Thái Linh	20215414	Linh.TT215414@sis.hust.edu.vn
2	Nguyễn Hoàng Lâm	20210551	Lam.nh210515@sis.hust.edu.vn
3	Phạm Đăng Khuê	20215406	Khue.pd215406@sis.hust.edu.vn
4	Trần Minh Chiến	20215321	Chien.tm215321@sis.hust.edu.vn
5	Lục Minh Hoàng	20215379	Hoang.lm215379@sis.hust.edu.vn

Giảng viên hướng dẫn: TS. Trần Việt Trung

Hà Nội, tháng 12 năm 2024

MỤC LỤC

I. Đặt vấn đề

Mô tả bài toán:
Lichess là một nền tảng cờ vua trực tuyến, nơi người dùng có thể chơi các trận đấu, xem lại các ván cờ, và phân tích các trận đấu. Dữ liệu từ Lichess bao gồm thông tin về các trận đấu cờ vua, các ván cờ, thông tin của người chơi, các nước đi, kết quả, và các thuộc tính khác. Dữ liệu này có thể rất lớn, với hàng triệu trận đấu và thông tin chi tiết cần được lưu trữ và xử lý.

1. Phân tích tính phù hợp của bài toán với Big Data

1.1. Volume (Khối lượng dữ liệu)

- Lichess chứa dữ liệu về hàng triệu trận đấu, mỗi trận đấu bao gồm nhiều thông tin chi tiết như lịch sử các nước đi, người chơi, kết quả trận đấu, Elo của người chơi, vv. Mỗi trận đấu có thể có đến vài trăm nước đi, tạo ra một lượng dữ liệu lớn cần được lưu trữ và xử lý.
- Khối lượng dữ liệu: Với hàng triệu trận đấu và các trận đấu mới diễn ra mỗi ngày, tổng lượng dữ liệu có thể lên đến terabytes hoặc thậm chí petabytes theo thời gian.
- Lưu trữ: Để xử lý và lưu trữ khối lượng dữ liệu này, các hệ thống như Cassandra, Hadoop HDFS hoặc Cloud Storage là cần thiết, vì chúng có khả năng mở rộng để đáp ứng yêu cầu về lưu trữ dữ liệu lớn.

1.2. Variety (Đa dạng dữ liệu)

- Dữ liệu từ Lichess không chỉ đơn giản là các trận đấu, mà còn bao gồm nhiều loại thông tin khác nhau:
    - Dữ liệu cấu trúc: Thông tin trận đấu (người chơi, kết quả, Elo, vv).
    - Dữ liệu bán cấu trúc: Các nước đi trong trận đấu, có thể được lưu trữ dưới dạng chuỗi (chẳng hạn như PGN).
    - Dữ liệu phi cấu trúc: Những nhận xét của người chơi, lịch sử trò chơi, vv.
- Sự đa dạng này đòi hỏi phải có một hệ thống có thể xử lý và phân tích các loại dữ liệu khác nhau. Hadoop và Spark có thể xử lý tốt dữ liệu dạng cấu trúc và phi cấu trúc, đồng thời Cassandra hỗ trợ lưu trữ các dữ liệu bán cấu trúc.

1.3. Velocity (Tốc độ dữ liệu)

- Dữ liệu thời gian thực: Các trận đấu đang diễn ra trên Lichess tạo ra một dòng dữ liệu liên tục cần được xử lý ngay lập tức. Các trận đấu có thể kéo dài vài giờ, và mỗi nước đi là một sự kiện cần được ghi lại và phân tích.
- Dữ liệu stream và batch: Cần có khả năng xử lý dữ liệu stream (khi trận đấu đang diễn ra, sử dụng Kafka và Spark Streaming) và dữ liệu batch (khi xử lý và phân tích các trận đấu đã kết thúc, sử dụng Spark Batch và Hadoop). Việc xử lý các dòng dữ liệu nhanh chóng giúp cung cấp các phân tích tức thời (như thống kê Elo, dự đoán kết quả trận đấu, vv.).

1.4. Veracity (Độ chính xác và độ tin cậy)

- Chất lượng dữ liệu: Dữ liệu từ Lichess có thể không phải lúc nào cũng chính xác 100% (ví dụ: lỗi trong việc ghi lại các nước đi, sai sót trong việc nhập thông tin người chơi), điều này có thể ảnh hưởng đến các phân tích và dự báo. Cần có các cơ chế kiểm tra và làm sạch dữ liệu trong quá trình xử lý.
- Độ tin cậy: Hệ thống cần phải đảm bảo rằng dữ liệu được ghi nhận chính xác từ nguồn và được lưu trữ một cách an toàn. Các cơ chế kiểm tra tính toàn vẹn dữ liệu cần được áp dụng trong hệ thống lưu trữ và xử lý, ví dụ như sử dụng Cassandra với cơ chế đảm bảo độ tin cậy cao.

2. Phạm vi và giới hạn của project

2.1. Phạm vi:

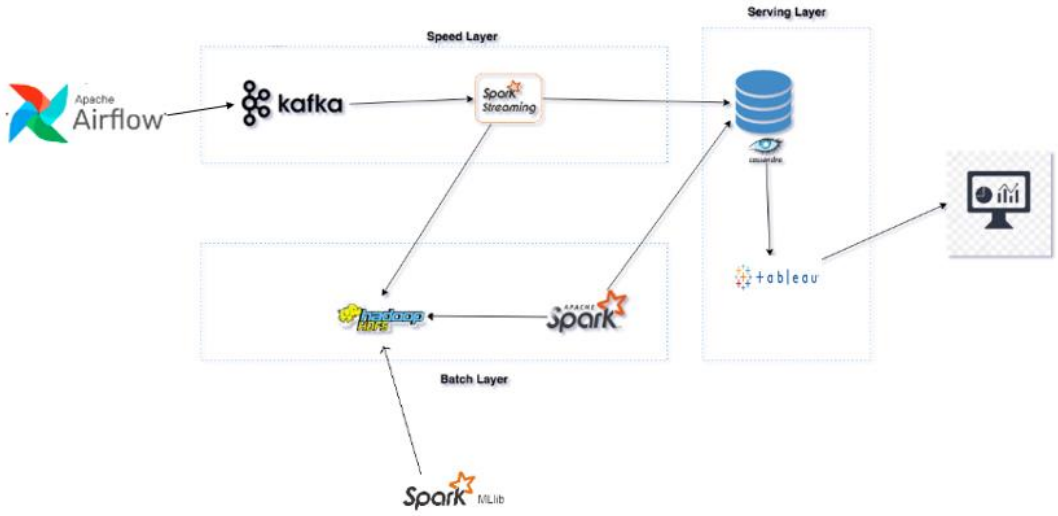
Dự án này sẽ bao gồm việc tải dữ liệu từ Lichess, lưu trữ vào cơ sở dữ liệu phân tán (Cassandra hoặc HDFS), xử lý và phân tích dữ liệu, và cung cấp API để truy vấn và phân tích dữ liệu.

2.2. Giới hạn:

Dự án sẽ không bao gồm việc phát triển giao diện người dùng hoặc các ứng dụng frontend. Tập trung vào xử lý dữ liệu, lưu trữ và cung cấp các API để truy vấn dữ liệu.

II. Kiến trúc và thiết kế

1. Kiến trúc tổng thể



Kiến trúc được sử dụng là kiến trúc Lamda:

- Batch Layer: Lưu trữ và xử lý dữ liệu từ Lichess trong Cassandra
- Speed Layer: Xử lý dữ liệu trong thời gian thực sử dụng Kafka và Spark Streaming để thu thập và xử lý các trận đấu diễn ra hiện tại.
- Serving Layer: Cung cấp các truy vấn từ dữ liệu đã xử lý và phân tích cho người dùng qua REST API.

2. Chi tiết từng Component và vai trò:

2.1. Dữ liệu:

Dữ liệu được lấy từ API do lichess cung cấp, bao gồm thông tin về các trận đấu cờ vua người chơi, kết quả trận đấu, lịch sử các nước đi, Elo, thời gian chơi…

Định dạng: Dữ liệu được lưu dưới định dạng PGN (Portable Game Notation). Các file PGN chứa thông tin chi tiết về các nước đi trong trận đấu.

Ví dụ về dữ liệu của nhiều trận đấu trong một file:

GameID	White	Black	Result	Date	Time	WhiteElo	BlackElo	TimeCont	Variant	Moves
0	cecht	molsonha	0-1	2024.09.01	0:25:35	1846	1967	60+0	Chess960	e2e4 h8g6 b2b3 b ..
1	BBBGirl	Ivo-Marin	1-0	2024.09.01	0:25:40	1894	1655	120+0	Chess960	e2e4 b7b6 d2d4 a ..
2	Magna19	AspectOf1	0-1	2024.09.01	0:25:40	1152	1966	120+0	Chess960	b2b4 e7e5 c2c4 f8 ..
3	vladimirzo	chessleo9	1-0	2024.09.01	0:26:18	1567	1566	300+0	Chess960	a1b3 d7d5 c2c3 c7 ..
4	Lary_Lodi	kuraftinhc	0-1	2024.09.01	0:26:24	1889	1915	90+0	Chess960	e1f3 a7a5 f3g5 e8 ..
5	Killda	gmweinbe	0-1	2024.09.01	0:26:54	1551	1616	600+0	Chess960	b2b4 g8f6 c1b2 d7 ..
6	nathantm	mle	0-1	2024.09.01	0:28:05	1970	1859	180+0	Chess960	c2c4 g8f6 b2b3 b7 ..
7	drogba13	Demoleit	0-1	2024.09.01	0:28:15	2070	1987	120+0	Chess960	e2e3 e7e5 c2c4 g8 ..
8	a_true_Ni	SLABIFOR	0-1	2024.09.01	0:28:15	2048	1860	120+0	Chess960	g2g3 a8b6 f2f4 d7 ..
9	Jack639	m1ntaka	0-1	2024.09.01	0:28:33	1268	1505	600+3	Chess960	c1d3 f7f6 e1d1 e7 ..
10	galeono	Tucapel	1-0	2024.09.01	0:36:00	1560	1634	300+3	Chess960	d2d4 d7d5 b1c3 c7 ..
11	Sapolsky	elvor1on	1-0	2024.09.01	0:36:11	1433	1432	240+0	Chess960	b2b3 e7e5 d2d4 c6 ..
12	Ivo-Marin	BBBGirl	1-0	2024.09.01	0:37:30	1663	1900	120+0	Chess960	g2g3 d8c6 h1g2 e7 ..
13	Demoleit	emadov77	1-0	2024.09.01	0:37:30	1987	2045	120+0	Chess960	f2f4 f7f5 g2g3 g7g ..
14	patolouco	Trasric	1-0	2024.09.01	0:38:15	1944	1689	180+0	Chess960	e2e4 c7c5 h1g3 b7 ..
15	mle	nathantm	0-1	2024.09.01	0:38:19	1859	1970	180+0	Chess960	e1d3 f7f5 d3e1 g7 ..
16	AspectOf1	Magna19	1-0	2024.09.01	0:38:47	1944	1149	120+0	Chess960	d1c3 c8d6 e2e4 d8 ..
17	SLABIFOR	Leandro85	0-1	2024.09.01	0:38:47	1864	2028	120+0	Chess960	e2e4 g7g6 g2g3 e8 ..
18	m1ntaka	BIN-1960	0-1	2024.09.01	0:32:07	1507	1604	600+3	Chess960	e2e4 e7e5 g1f3 d7 ..
19	mle	nathantm	0-1	2024.09.01	0:32:44	1867	1962	180+0	Chess960	c2c4 c7c5 b2b3 g8 ..

2.2. HDFS

Chức năng:

- HDFS là hệ thống tệp phân tán của Hadoop, giúp lưu trữ và quản lý dữ liệu lớn một cách phân tán trên các máy chủ.
- Dữ liệu được chia thành các khối nhỏ (blocks) và phân tán trên nhiều nút trong hệ thống, đảm bảo độ tin cậy và khả năng chịu lỗi.

Lợi ích:

- Lưu trữ dữ liệu lớn: HDFS được thiết kế để lưu trữ khối lượng lớn dữ liệu không cấu trúc hoặc bán cấu trúc, thích hợp cho các hệ thống yêu cầu xử lý dữ liệu quy mô lớn.
- Khả năng chịu lỗi: Dữ liệu được sao lưu nhiều lần trên các nút khác nhau, giúp phục hồi dữ liệu khi xảy ra sự cố với một hoặc nhiều nút.
- Mở rộng dễ dàng: HDFS có khả năng mở rộng theo chiều ngang, dễ dàng thêm nút mới để tăng dung lượng lưu trữ mà không làm gián đoạn hoạt động.

Nhược điểm:

- Độ trễ cao: HDFS không phù hợp với các ứng dụng yêu cầu độ trễ thấp, vì việc đọc dữ liệu từ hệ thống tệp phân tán có thể mất thời gian.

- Yêu cầu cấu hình phức tạp: Cấu hình và duy trì một cụm HDFS có thể phức tạp, đặc biệt là khi hệ thống có quy mô lớn.

2.3. Kafka

Chức năng:

- Kafka là một hệ thống quản lý dòng sự kiện (event streaming platform), cung cấp khả năng xử lý dòng dữ liệu thời gian thực.
- Kafka giúp truyền tải dữ liệu giữa các thành phần của hệ thống, đảm bảo việc truyền tải tin nhắn, sự kiện hoặc dữ liệu qua các kênh (topics) một cách nhanh chóng và hiệu quả.

Lợi ích:

- Xử lý dữ liệu thời gian thực: Kafka giúp xử lý và phân phối dữ liệu theo thời gian thực, rất hữu ích cho các ứng dụng cần thu thập và phân tích dữ liệu ngay khi nó được tạo ra (như các trận đấu cờ vua đang diễn ra).
- Khả năng chịu lỗi cao: Kafka đảm bảo độ tin cậy cao, vì dữ liệu có thể được sao lưu và phục hồi khi xảy ra sự cố.
- Khả năng mở rộng: Kafka có thể xử lý một lượng lớn dữ liệu với khả năng mở rộng ngang và phân tán trên nhiều máy chủ.

Nhược điểm:

- Phức tạp trong việc triển khai và bảo trì: Cài đặt và bảo trì Kafka có thể phức tạp, đặc biệt là khi hệ thống có quy mô lớn với nhiều kênh dữ liệu.
- Không hỗ trợ tính toán phân tán phức tạp: Kafka chủ yếu xử lý việc truyền tải dữ liệu và không hỗ trợ các phép toán phức tạp, như tính toán hay phân tích dữ liệu trực tiếp.

2.4. Spark

Chức năng:

- Apache Spark là một nền tảng xử lý dữ liệu phân tán, hỗ trợ xử lý dữ liệu theo cả chế độ batch và stream.
- Spark cung cấp một số thư viện mạnh mẽ cho phân tích dữ liệu, học máy, và xử lý dữ liệu thời gian thực.

Lợi ích:

- Xử lý dữ liệu nhanh: Spark có khả năng xử lý dữ liệu nhanh hơn so với Hadoop MapReduce nhờ vào việc sử dụng bộ nhớ RAM để lưu trữ dữ liệu tạm thời.
- Hỗ trợ xử lý dữ liệu theo dòng và theo lô: Spark có thể xử lý dữ liệu theo dòng (streaming) hoặc theo lô (batch), giúp hệ thống linh hoạt trong việc xử lý các dữ liệu khác nhau.
- Tích hợp dễ dàng với nhiều hệ thống lưu trữ: Spark có thể kết nối và tích hợp với các hệ thống lưu trữ như Cassandra, HDFS, và các cơ sở dữ liệu quan hệ.

Nhược điểm:

- Yêu cầu tài nguyên cao: Spark yêu cầu nhiều tài nguyên tính toán, đặc biệt khi xử lý dữ liệu lớn hoặc phức tạp. Việc tối ưu hóa tài nguyên có thể trở thành thách thức.
- Phức tạp trong việc tối ưu hóa: Spark có thể gặp khó khăn khi tối ưu hóa các tác vụ tính toán phức tạp, đặc biệt khi dữ liệu lớn và không thể lưu trữ trong bộ nhớ.

2.5. Cassandra

Chức năng:

- Cassandra là một hệ quản trị cơ sở dữ liệu NoSQL, cung cấp khả năng lưu trữ dữ liệu phân tán và có thể mở rộng linh hoạt.
- Dữ liệu được lưu trữ dưới dạng bảng, với khả năng phân tán trên nhiều nút (nodes) trong cụm (cluster).
- Cassandra được tối ưu hóa cho việc ghi và đọc với tốc độ cao, thích hợp cho các ứng dụng yêu cầu xử lý dữ liệu lớn trong môi trường phân tán.

Lợi ích:

- Khả năng mở rộng linh hoạt: Cassandra hỗ trợ khả năng mở rộng ngang (horizontal scaling), giúp dễ dàng mở rộng hệ thống bằng cách thêm các nút mới mà không làm gián đoạn hoạt động của hệ thống.
- Khả năng chịu lỗi: Với cơ chế sao lưu và phân tán dữ liệu, Cassandra cung cấp tính năng phục hồi nhanh chóng khi một nút gặp sự cố.
- Tốc độ ghi cao: Cassandra có thể xử lý hàng triệu thao tác ghi mỗi giây, rất phù hợp với các ứng dụng cần xử lý dữ liệu nhanh chóng, như lưu trữ các trận đấu cờ vua trong hệ thống.

Nhược điểm:

- Khó khăn trong việc thực hiện các truy vấn phức tạp: Cassandra không hỗ trợ JOIN và các truy vấn phức tạp như các hệ quản trị cơ sở dữ liệu quan hệ, khiến việc thực hiện các phép toán liên kết dữ liệu trở nên khó khăn hơn.
- Không hỗ trợ các tính toán ACID đầy đủ: Cassandra chỉ hỗ trợ các tính toán BASE (Basically Available, Soft state, Eventual consistency), vì vậy việc đảm bảo tính nhất quán dữ liệu trong các tình huống phức tạp có thể gặp khó khăn.

2.6. Airflow

Chức năng:

- Apache Airflow là một công cụ mã nguồn mở dùng để tự động hóa, lập kế hoạch và giám sát các quy trình công việc (workflows). Airflow cho phép định nghĩa, lên lịch và theo dõi các quy trình xử lý dữ liệu phức tạp, giúp việc xử lý và di chuyển dữ liệu giữa các hệ thống trở nên dễ dàng hơn.
- Airflow hỗ trợ mô hình lập trình DAG (Directed Acyclic Graph), trong đó các task được kết nối với nhau theo các mối quan hệ phụ thuộc, từ đó đảm bảo các công việc được thực hiện theo đúng thứ tự.

Lợi ích:

- Tự động hóa quy trình công việc: Airflow giúp tự động hóa các quy trình dữ liệu phức tạp, như ETL (Extract, Transform, Load), giúp giảm thiểu công sức quản lý thủ công và tránh sai sót trong quá trình thực hiện các tác vụ.
- Quản lý và giám sát dễ dàng: Airflow cung cấp giao diện người dùng (UI) giúp theo dõi và giám sát tình trạng các task, giúp các nhà quản trị dễ dàng xác định và xử lý sự cố khi có lỗi xảy ra trong quy trình công việc.
- Tính linh hoạt cao: Airflow có thể tích hợp với nhiều công cụ và dịch vụ khác nhau, bao gồm các hệ thống lưu trữ (như HDFS, Cassandra), công cụ phân tích (như Spark), và công cụ xử lý luồng (như Kafka).
- Khả năng mở rộng và phân tán: Airflow có thể dễ dàng mở rộng bằng cách thêm các worker để xử lý các quy trình công việc nặng, giúp hệ thống có thể đáp ứng yêu cầu quy mô lớn.

Nhược điểm:

- Cấu hình phức tạp: Việc thiết lập và cấu hình một môi trường Airflow có thể trở nên phức tạp, đặc biệt khi hệ thống có nhiều quy trình công việc và cần phải tích hợp với các dịch vụ khác.
- Tài nguyên yêu cầu: Airflow cần nhiều tài nguyên để chạy hiệu quả, đặc biệt khi có nhiều task cần thực hiện đồng thời. Việc quản lý và tối ưu hóa tài nguyên có thể là một thách thức trong các hệ thống quy mô lớn.
- Quản lý các task phức tạp: Mặc dù Airflow hỗ trợ xử lý các task phức tạp, nhưng việc duy trì các DAG lớn và phức tạp có thể gặp khó khăn và dễ dẫn đến các lỗi không mong muốn nếu không được quản lý cẩn thận.

2.7. Lợi ích trong việc tích hợp Airflow vào trong việc phân tích dữ liệu cờ vua:

Tự động hóa các quy trình dữ liệu:

- Với các quy trình như tải dữ liệu từ Lichess API, xử lý dữ liệu, và lưu trữ vào các hệ thống như Cassandra hoặc HDFS, Airflow có thể tự động hóa hoàn toàn các công việc này, đảm bảo tính nhất quán và hiệu quả trong việc di chuyển và xử lý dữ liệu.

Giám sát quy trình dữ liệu:

- Airflow giúp theo dõi trạng thái của từng task trong quy trình, ví dụ như việc kiểm tra tình trạng của các trận đấu cờ vua đang được xử lý. Nếu có sự cố xảy ra trong quá trình tải hoặc xử lý dữ liệu, Airflow sẽ cảnh báo để người quản trị có thể can thiệp kịp thời.

Lên lịch và điều phối quy trình công việc:

- Airflow cho phép lên lịch tự động cho các quy trình như xử lý dữ liệu theo lô hoặc cập nhật dữ liệu theo thời gian thực. Điều này đặc biệt hữu ích khi dữ liệu từ Lichess liên tục thay đổi và cần được xử lý và phân tích liên tục.

Khả năng tích hợp với các hệ thống khác:

- Airflow có thể tích hợp với Kafka để xử lý dữ liệu stream, với Spark để phân tích dữ liệu, và với Cassandra/HDFS để lưu trữ dữ liệu. Việc tích hợp này giúp hệ thống trở nên linh hoạt và có thể xử lý được mọi yêu cầu dữ liệu lớn trong ứng dụng cờ vua.

3. Data flow và component interaction diagrams

Data Ingestion:

- Lichess Database→ Kafka → Spark Streaming → Cassandra (lưu trữ dữ liệu thời gian thực)

Batch Processing:

- Lichess API → Spark Batch → Cassandra (lưu trữ dữ liệu đã xử lý)

Truy vấn và Phân tích:

- Người dùng → REST API → Cassandra (truy vấn dữ liệu trận đấu, Elo, vv)

3.1 Luồng dữ liệu từ lichess Database → Kafka

- Lichess cung cấp API cho phép truy xuất thông tin về các trận đấu cờ vua. Dữ liệu về một trận đấu có thể được cung cấp dưới dạng PGN (Portable Game Notation), đây là định dạng chuẩn để lưu trữ thông tin về một ván cờ vua, bao gồm các nước đi, tên người chơi, kết quả trận đấu, thời gian, và các thông tin liên quan khác.
- Dữ liệu PGN có thể được truy xuất qua một endpoint API của Lichess, ví dụ như:
- - Endpoint Lichess API: https://database.lichess.org/antichess/lichess_db_antichess_rated_2024-11.pgn.zst
- Dữ liệu trả về: Một hoặc nhiều trận đấu được lưu trữ dưới dạng PGN

```
[Event "Rated Antichess tournament https://lichess.org/tournament/nw1fDm9o"]
[Site "https://lichess.org/zRcTgBRK"]
[Date "2024.09.01"]
[Round "-"]
[White "guazun"]
[Black "Hitskie"]
[Result "1-0"]
[UTCDate "2024.09.01"]
[UTCTime "00:22:04"]
[WhiteElo "2140"]
[BlackElo "2110"]
[WhiteRatingDiff "+5"]
[BlackRatingDiff "-6"]
[TimeControl "180+0"]
[Termination "Normal"]
[Variant "Antichess"]

1. e3 { [%clk 0:03:00] } 1... c5 { [%clk 0:03:00] } 2. b4 { [%clk 0:02:58] } 2... cxb4 { [%clk 0:02:59] } 3. Ba6 { [%clk 0:02:58] } 3...
```

-
- Xử lý dữ liệu PGN
- Sau khi dữ liệu PGN được lấy từ Lichess API, nó sẽ cần được xử lý để chuẩn hóa bằng cách chuyển đổi thành định dạng JSON rồi tiến hành gửi vào kafka thông qua một kafka producer. Kafka Producer sẽ chịu trách

nhiệm đẩy dữ liệu vào các topic Kafka mà các consumer có thể xử lý sau này.

- Quy trình đẩy dữ liệu vào Kafka:
  Tạo một kafka producer để gửi các thông điệp dữ liệu vào một topic kafka, topic name: “lichess_game”
- Topic sẽ chứa thông điệp là các trận đấu cờ vua dưới dạng JSON đã được chuẩn hóa từ PGN. Mỗi thông điệp sẽ là một trận đấu.
- Kafka commer sẽ lắng nghe topic này và xử lý dữ liệu theo yêu cầu.

3.2. Luồng dữ liệu từ Kafka → Spark → Cassandra

- Mục tiêu: Spark sẽ là công cụ xử lý dữ liệu (streaming hoặc batch) để phân tích, tính toán các chỉ số hoặc trích xuất thông tin quan trọng từ dữ liệu PGN về các trận đấu cờ vua trước khi lưu vào Cassandra
- Quy trình:

  Kafka Consumer là nguồn dữ liệu đầu vào, spark streaming sẽ lắng nghe các topic Kafka, ở đây sử dụng topic “líchess_game” nơi các trận đấu cờ vua được đẩy vào dưới dạng JSON.

  Spark Streaming thực hiện công việc transform dữ liệu bằng các API được cung cấp:

  - filter: lọc thông tin tên người chơi, Elo, kết quả trận đấu, thời gian và các nước đi.

  - select: chọn cột lấy thông tin cần thiết

  - groupBy : tổng hợp kết quả phân tích

  Lưu dữ liệu vào Cassandra

  Bảng trong Cassandra sẽ được lưu trữ thông tin về các trận đấu cờ vua:

Tên cột	Kiểu dữ liệu	Mô tả
Gameid	UUID	Khóa chính bảng games
Event	Text	Tên sự kiện
Date	Datatime	Ngày diễn ra trận đấu
Players.white.name	Text	Tên người chơi trắng

Players.black.name	Text	Tên người chơi đen
Players.white.whileElo	Text	Số Elo người chơi trắng
Players.black.blackElo	Text	Số Elo người chơi đen
Result	Text	Kết quả trận đấu
Moves	Text	Danh sách nước đi

3.3. Lichess Database → Spark Batch → Cassansdra

- Quy trình:
- Dữ liệu từ Lichess được cung cấp thông qua các API công khai, như API lichess game data dưới dạng PGN. Spark sẽ sử dụng thư viện Requests trong python hoặc HTTP trong scala để gửi yêu cầu get tới Lichess API và lấy dữ liệu và chuyển đổi về dạng parquet lưu trong HDFS.
- Spark đọc dữ liệu JSON từ HDFS, phân tích thông tin trận đấu như Elo, nước đi, kết quả, số trận thắng/thua và lưu vào Cassandra.

III. Chi tiết triển khai

https://github.com/khued200/Lichess-BigData-Processing.git

IV. Bài học kinh nghiệm

1. Khả năng chịu lỗi với kafka broker
   - Context: Hệ thống xử lý file dữ liệu lớn từ database và gửi vào kafka thông qua các container trong thời gian dài
   - Thách thức: Lượng dữ liệu lớn nhanh chóng nếu không được consume sẽ vượt quá khả năng lưu trữ và xử lý của hệ thống, và nếu có thể mất dữ liệu nếu broker chết(bị tắt rồi khởi động lại)
   - Impact: Hệ thống trở nên chậm chạp, không thể xử lý thêm dữ liệu và thậm chí bị "crash". Gây ra mất mát đữ liệu khi broker chết và khởi động lại
   - Giải pháp đã sử dụng : Thêm nhiều broker chứa nhiều partition của topics giúp có thể khôi phục và tránh mất mát dữ liệu.
2. Thực hiện song song các task của DAG

- Context: DAG gồm xử lý file gồm 2 công đoạn: process và send đến kafka. Trường hợp dữ liệu quá lớn phải thực hiện lần lượt process, xong mới đến send data
- Thách thức: Lượng dữ liệu lớn cần xử lý thành rồi mới thực hiện send data. Nếu chưa send hết data mà broker gặp lỗi sẽ phải thực hiện retry, gây tốn kém tài nguyên và thời gian.
- Impact: Hệ thống trở nên chậm chạp, không thể xử lý thêm dữ liệu và thậm chí bị "crash". Gây ra tình trạng tắc nghẽn dữ liệu.
- Giải pháp đã sử dụng : Chia thành các luồng thực hiện song song process và send_data. Mỗi luồng sẽ chịu trách nhiệm với một đoạn của dữ liệu. Nếu một broker tắt, sẽ chỉ ảnh hưởng một vài luồng.

3. Khả năng chịu lỗi của task DAG

- Context: DAG gồm xử lý file gồm 2 công đoạn: process và send đến kafka. Nếu đang trong công đoạn send_data mà một(nhiều) broker chết vĩnh viễn.
- Thách thức: Các luồng do broker chết chịu trách nhiệm sẽ không có khả năng gửi dữ liệu đến kafka.
- Impact: Hệ thống mất mát dữ liệu ở các luồng mà do broker đó chịu trách nhiệm.
- Giải pháp đã sử dụng : Thêm 1 task `retry_task` chịu trách nhiệm thu thập các data của các luồng bị mất broker, tìm kiếm broker còn hoạt động, gửi các dữ liệu còn thiếu đến broker đó.

V. Kết luận

Trong dự án này, nhóm đã thành công trong việc xây dựng một hệ thống xử lý và phân tích dữ liệu cờ vua. Các kết quả đạt được qua quá trình thực hiện dự án:

- Xây dựng thành công hệ thống sử dụng các công nghệ bigdata với Airflow, Kafka, Spark, Cassandra.
- Thiết lập dataflow từ airflow --> kafka --> spark --> cassandra.
- Thử nghiệm khả năng chịu lỗi của kafka, airflow với các tình huống.